Replica Divergence Control Protocol Based on Predicted Profile

Ahmed Jebali, Mesaac Makpangou

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Replica Divergence Control Protocol Based on Predicted Profile

Ahmed Jebali — Mesaac Makpangou

N° 4386

February 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4386--FR+ENG

INRIA
ROCQUENCOURT

Replica Divergence Control Protocol Based on Predicted Profile

Ahmed Jebali *, Mesaac Makpangou †

Thème 1 — Réseaux et systèmes
Projet SOR

Rapport de recherche n° 4386 — February 2002 — 17 pages

Abstract: The Divergence Control Protocol Based on Predicted Profile allows cooperative processes deployed on a weakly-connected environment to conciliate the offering of low response time to clients with the guarantee of a bounded consistency, despite characteristics of the underlying communication infrastructure. This report presents how to decompose a global divergence bound into a set of local criteria that are locally checked at any invocation. We propose a protocol relying on these criteria to incrementally construct histories that preserve at any time, the global divergence bound between any replica and the "ideal state" of the replicated object. An implementation of the protocol is discussed. This report reports a number of experiments conducted in order to evaluate the rationale and the performance of the proposed protocol. These experiments show that the protocol benefits a lot from the provision of a correct profile and outperforms intuitive cooperation protocols in large-scale settings.

Key-words: Distributed System, Network Service, Replication Protocol, Divergence Control, Bounded Consistency.

* Ahmed.Jebali@inria.fr
† Mesaac.Makpangou@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

Protocole de Contrôle de Divergence Basé sur la Prédiction de Profil

Résumé : Le protocole de contrôle de divergence basé sur la prédiction de profile permet à des processus déployés sur un environnement faiblement connecté de concilier performance et cohérence des données, malgré les caractéristiques inhérentes à ce type d'environnements. Ce rapport présente comment décomposer un critère de cohérence globale en un ensemble de critères locaux, qui sont vérifiés à chaque invocation. Le protocole proposé construit incrémentalement, en utilisant ces critères, les histoires qui préservent à tout instant le critère de divergence globale entre chaque réplicat et le réplicat «idéal». Nous décrivons ici une implantation de ce protocole, et discutons ensuite ces performances. Les mesures de performance montrent que le protocole est adapté aux environnements large échelle.

Mots-clés : Systèmes distribués, service distribué, protocole de réplication, contrôle de divergence, divergence bornée

1 Introduction

Many applications world-wide accessed can be replicated over the Internet to improve their performance and their availability. However, replication in large-scale settings raises the issue of the consistency of conflicting accesses concurrently performed on different replicas. The difficulty stems from the characteristics of network: communication failures, disconnections and partitions. Protocols capable of ensuring a global ordering [BHG87] achieve poor performance.

Fortunately, all applications do not need strong consistency. In particular, for a wide range of applications, the correctness of the services delivered to clients requires only that the divergence between replicas remains under some limit. An example of such applications is a distributed load balancing system implemented by a group of cooperative managers. Each manager handles requests from its local clients, based on its local view of the load on different resources. It is clear that the information available to each manager does not represent the instantaneous state of the system, rather an approximation of that state. Of course, a good load balancing system must be able to make correct (i.e., acceptable) decision from the approximate state available to it. It is also clear that, if the error between the real state and the ones available to managers is arbitrary large, it becomes too difficult to any system to make sensible placement decision. Other examples of applications tolerating some amount of error include poll system, statistics supply systems, and data warehouses.

In the past, a number of proposals [ABM90, PL91, YV00b] were made to exploit this tolerance to improve the performance and the availability of these applications, while guaranteeing the correctness of the service delivered to clients. However, none of these previous works addresses the issue when replicas are deployed on a weakly-connected environment. With the rapid development of the mobile computing and a continuous success of the Web, we anticipate that replicas of most applications world-wide accessed will not be accessible all the time. Hence, each replica needs a way to determine, without communicating with its peers, operations that they have initiated. One interesting solution is to rely on the profile of the application at each site. On the one hand, most concerned applications run for long periods of time. On the order hand, a number of tools that could help collect application historic exist. This historic can be used to predict the behaviour of a process, thanks to data analysis and data mining techniques [Vap98].

This paper presents a bounded divergence control protocol that exploits the application profile to augment the autonomy of each replica. Roughly, each replica is passed the application profile at each site, as well as a consistency criterion that it checks at any request. The evaluation requires only local information. If the check succeeds, the operation is allowed without any synchronization. If at some site, the actual behaviour diverges too much from the announced profile, this site initiates a profile update. This requires a global synchronization with all replicas.

The rest of the paper is organized as follows. Section 2 presents the replication model. Section 3 provides a formal definition of the divergence, then discusses how to capture the divergence between the real behaviour and the one predicted by the announced profile, and finally it presents the decomposition of the global consistency criterion into a tuple

of sufficient local conditions. In section 4 we give the algorithm performed at each site and present its implementation on top of the Ensemble group communication platform. Section 5 discusses the experiments we conducted in order to evaluate the performance and the merit of the proposed protocol with respect to other cooperative protocols. Section 6 presents considerations on how to determine the consistency bound for a given application. It provides also some hints on the characteristics of the targeted applications. Section 7 compares our proposal with related work. Finally, section 8 draws some conclusions and presents our perspectives.

2 Replication Model

We consider distributed applications composed of groups of replicated processes noted $(p_1, \ldots, p_n)$ and located on different sites noted $(S_1, \ldots, S_n)$. These processes cooperate through a shared replicated object. There is one replica of the shared object on each site. Each site is associated with a consistency manager that controls operations performed on the local replica of the shared object. An *operation* is an event chain that leads to a particular access to the shared object.

Each process accesses the shared replicated object by initiating operations, via the local consistency manager. Each operation initiated by one process is eventually propagated to all consistency managers attached to replicas of the shared object. Each manager decides the order in which its associated replica of the shared object performs operations initiated both locally and by remote process. The primary role of a consistency manager is to make sure that new operations can be initiated by the local process without compromising application consistency. For that, each consistency manager is parameterized with an applicative consistency criterion that is evaluated when the local process wants to initiate a new operation. If the evaluation succeeds, the requested operation is accepted and can therefore be scheduled on this site regardless the activity on other sites. Otherwise, a synchronization with other consistency managers is required.

The synchronization allows consistency managers to reduce the divergence between their local views of the application's state and the real one (that is the set of operations initiated so far by the group of processes composing the application). Note that the model doesn't specify the way operations initiated on one site happen to be seen by the consistency manager located on a different site, nor the acknowledgement mechanism. One can use the anti-entropy protocol [PST+97], gossip messages [LLSG92], or any reliable propagation protocol. A more detailed description of the replication model can be found in [BC98, JM01a, JM01b].

3 Bounding Replica Divergence

3.1 Divergence Definition

A replica is fully determined by the sequence of operations that it executes. The shared object is the real object obtained by executing all operations initiated on all replicas. Thus

the divergence is the difference between the replica and the real shared object. This difference depends on the operations locally accepted on replicas and not propagated to others. The difficulty is how to determine these operations without continuously communicating with other replicas.

Let us assume that the sequence of operations which a process p_i is likely to initiate during the application's lifetime is characterized by a profile function denoted $\mathcal{P}_i$. The profile maps to each time interval the history of operations that are expected from process p_i during that time interval [1].

The idea is to allow consistency managers to use profile functions to approximate remote process histories, without having to communicate with them, while still guaranteeing application consistency. Since prediction is not always accurate, these approximations could diverge from the actual behaviours of processes. The more accurate is the prediction provided by the profile functions, the smaller is that divergence. Our goal is to provide consistency managers with a way to bound this divergence to a value B.

3.2 Bounding the Divergence

We note $I_i(t',t) = (op_1, op_2, \ldots, op_k)$ the ordered sequence of operations initiated by process p_i during the time interval $[t',t]$. We simply note $I_i(t)$ if t' is the start time of the applications. With respect to a process $p_j, j \neq i$, the history of p_i is composed of two parts: (1) $A_{i,j}$ the sequence of operations initiated by p_i and already acknowledged by site S_j and (2) $W_{i,j}$ the sequence of operations initiated by p_i yet to be propagated to the consistency manager on site S_j or yet to be acknowledged by that consistency manager. Using a concatenation operator $\oplus$ [2] on histories, we have the following equality: $I_i(t) = A_{i,j}(t') \oplus W_{i,j}(t',t)$.

We attach to an operation *op* a measure $\mu(op)$ that represents the impact of its application on the shared object. For example, the measure of $get(amount)$ on a stock is $(-amount)$ since its impact is to decrement the stock by the value $amount$. The measure of a history is the sum of measures of operations of this history. For a given history $\mathcal{H} = (op_1, op_2, \ldots, op_k)$ the measure is: $\mathcal{M}(\mathcal{H}) = \mu(op_1) + \mu(op_2) + \ldots + \mu(op_k)$.

Consider now the tuple RH that represents the real histories that actually occurred on processes $(p_1, \ldots, p_n)$;

$$RH(t) = (I_1(t), I_2(t), \ldots I_n(t))$$

A consistency manager on site S_i approximates this global state with EH_i as follows:

$$EH_i(t) = (A_1(T_i[1]) \oplus \mathcal{P}_1(T_i[1],t), \ldots, I_i(t), \ldots,$$

$$A_n(T_i[n]) \oplus \mathcal{P}_n(T_i[n],t))$$

where $T_i[j]_{1 \leq j \leq n}$ is the occurrence date of the last operation initiated by process p_j and already acknowledged by p_i; $\mathcal{P}_j$ is the profile function of process p_j. We define the divergence

[1] We will see later that only the impact of predicted operations is used in the protocol.

[2] The $\oplus$ operator concatenates histories. $h_1 \oplus h_2$ is the history formed by operations of h_1 followed by operations of h_2.

(d_i) between RH and EH_i as follows:

$$d_i = |\mathcal{M}(I_1(t), I_2(t), \ldots I_n(t)) - \mathcal{M}(A_1(T_i[1]) \oplus \mathcal{P}_1(T_i[1], t), \ldots, I_i(t), \ldots, A_n(T_i[n]) \oplus \mathcal{P}_n(T_i[n], t))|$$

rewritten to:

$$d_i = |\Sigma_{j \neq i}^{n} \mathcal{M}(I_j(t)) - \mathcal{M}(A_j(T_i[j]) \oplus \mathcal{P}_j(T_i[j], t))|$$

Since $I_j(t) = A_j(T_i[j]) \oplus W_j(T_i[j], t)$, the divergence is:

$$d_i = |\Sigma_{j \neq i}^{n} \mathcal{M}(W_j(T_i[j], t) - \mathcal{M}(\mathcal{P}_j(T_i[j], t))|$$

Using the properties of absolute values, the present equation implies that:

$$d_i \leq \Sigma_{j \neq i}^{n} |\mathcal{M}(W_j(T_i[j], t) - \mathcal{M}(\mathcal{P}_j(T_i[j], t))|$$

Hence, to guarantee that d_i is less than a fixed bound B, it suffices to find a tuple local bounds $(b_1, \ldots, b_n)$ such that:

$$\forall j,\ b_j \geq 0 \ and \ \Sigma_{j=1}^{n} b_j \leq B \tag{1}$$

$$\forall j,\ |\mathcal{M}(W_j(T_i[j], t) - \mathcal{M}(\mathcal{P}_j(T_i[j], t))| \leq b_j \tag{2}$$

To guarantee these conditions it suffices that each consistency manager constraints its local behaviour to be conformable to its profile. The protocol has just to enforce this local condition and propagates operations to its peers when needed.

4 Divergence Control Protocol

4.1 Algorithm

The protocol bounds the divergence of the actual history of each process with respect to approximations that are made by other partners. We consider a tuple $(b_i, \ldots, b_n)$ such that $\Sigma_{j=1}^{n} b_j \leq B$. We assign the value b_j to the consistency manager on site S_j. When a process p_j wants to execute an operation op at date t, it informs its consistency manager, which in turn performs the following algorithm:

1. If for all i, the condition $|\mathcal{M}(W_j(T_i[j], t) + \mu(op) - \mathcal{M}(\mathcal{P}_j(T_i[j], t)| \leq b_j$ holds then the operation is accepted and added to the local history.
2. Otherwise, a synchronization is required. The local manager sends to each partner k such that $|\mathcal{M}(W_j(T_k[j], t) + \mu(op) - \mathcal{M}(\mathcal{P}_j(T_k[j], t)| > b_j$, operations initiated since the last operation acknowledged by k.

A process can also send notification at any time; this is not necessary to ensure the divergence condition, but increases system performance. When receiving a notification message from p_j, p_k updates the corresponding entry in its notification date vector T_k and executes notified operations on its own replica.

```
let requestHandler origin properties =
let predictedEffect = (*calls profiles functions*)
 let effectiveEffect = (*calls a cumulative variable*)
 let error = abs (effectiveEffect + properties.measure - predictedEffect)
 if error > b (*the local bound*) then
   (*operation is accepted and executed locally*)
 else
   serverAsync; (*demands an immediate asynchronous call back to send a notification*)
Send (origin, Respond reqNum) (*application response*)
```

(a) Local interface

4.2 Protocol Implementation

The divergence control protocol is implemented by a group of cooperative consistency managers. Each consistency manager implements two interfaces, one to communicate with the local application and one to communicate with other consistency managers. Interfaces define a set of events handlers.

The local interface enables application processes, sharing the replicated object, to declare operations that they want to perform. The main handler of this interface is requestHandler(see Figure (a)). When application wants to initiate an operation it invokes this handler with its client number origin and operation properties. Properties include the measure and the semantic of the operation (current implementation performs two semantics: increase or decrease the value of the shared object). Consistency managers define two variables effectiveEffect and predictedEffect that corresponds respectively to $\mathcal{M}(W_j(T_i[j],t)$ and $\mathcal{M}(\mathcal{P}_j(T_i[j],t)$. Upon the reception of a client operation request the consistency manager computes the error that may be introduced by this operation. The operation is accepted only if this error does not exceed the local bound.

When the divergence condition does not hold, a notification is requested via an immediate call back from consistency manager interface. This second interface handles communications with other consistency managers. Figure (b) gives a list of its most important handlers.

When a consistency manager receives a message from its peers it calls the receive function. Handlers are called depending on the message type. The serverReadyHandler is called to initialize the consistency manager and instantiate the local replica. The notificationHandler is called when a notification is received. A notification includes the list of operations performed by the sender consistency manager and not yet acknowledged by other consistency managers. In our implementation it is reduced to the total effect of these operations. When a consistency manager receives a notification it adds these notified operations to its local history. The actionHandler is called when a member joins or leaves the group of consistency

```
let receive origin msg =
match msg with
 | ServerReady ->
    serverReadyHandler()
 | Notification ->
    notificationHandler()
let actionHandler
let heartBeatHandler
```

(b) Manager interface

managers; this handler is responsible of redistributing the global consistency bound B and reinitializing data structures to match the new group view. Finally the heartBeatHandler implements the notification sending function that is requested by the local interface.

Consistency managers communicate using an Ensemble stack [Hay98]. We use a reliable group communication stack composed of a virtual synchrony layer, an xfer layer and a local delivery layer. The xfer layer serves to handle safely the view changes when a consistency manager joins or leaves. We use the local delivery layer as an acknowledgment mechanism. That is, a message cast to the group and locally delivered is considered as acknowledged. This detail will be released in future implementation, by using a separate acknowledgement mechanism.

5 Performance Evaluation

To evaluate the relevance of our protocol performance, we developed a distributed electronic market space. This application is an interesting example of cooperation that could benefit from the divergence control protocol: the main issue for each site is to serve as many consumers as possible without causing the overall system to violate the consistency and without having to synchronize cooperative processes all the time. Roughly, we consider the case of a distribute community of producers and consumers that cooperate thanks to the service offered by a distributed electronic market space. Producers register their offers and consumers request resources. We developed the consistent distributed market space relying on the divergence control protocol as follows. Firstly, on each market site, we have one application process. Its role is to treat requests from the local consumers and producers. This process interacts with the consistency manager as described in Section 4.2. Secondly, on each market site, an observer process is running. It monitors the producer and the consumer profile. When a significant change is noticed, it initiates a procedure to update the current profile known to the group of consistency managers.

For the sake of simplicity, we make the following assumptions: (1) the number of sites where the electronic market is deployed is known and is equal to n; (2) all resources are equivalent, consequently the pool of resources can be represented by a numerical value corresponding to the number of resources available in the shared pool; (3) the behaviour of consumers and producers attached to site S_i can be approximated with two linear functions: $cons_i(t) = f_{cons_i} \times t$ and $prod_i(t) = f_{prod_i} \times t$. f_{cons_i} is the average arrival frequency of consumption requests on process p_i and f_{prod_i} is the average arrival frequency of production requests on process p_i.[3]

We run 10 allocators on 10 Pentium based PCs running Gnu/Linux. We run a gossip daemon on each machine to optimize message exchange. All experimentations are carried over 10 rounds for each configuration and the results presented later are the mean average of these rounds. The bound B is expressed in terms of resources; this corresponds to the maximum number of conflictual allocations that are tolerated.

We focused the evaluation on three goals :

- understanding the impact of prediction error on the protocol (section 5.1);
- evaluating the protocol in large scale settings (section 5.2) and
- comparing it to other cooperation protocols (section 5.3).

5.1 Profile Accuracy Impact

As the protocol relies on profiles prediction we anticipate that its performance is strongly related to the accuracy of such prediction. Thus we measured the performance of the protocol in two cases:

1. The perfect case: at each site the consumption and the production rate conform to the predicted profile;
2. Production miss case: consumers behave accordingly to their predicted profiles, but producers create less resources than predicted.

In the first experimental case, all consistency managers are notified of the profile of each allocator (that is the consumptions and productions that are expected from each site). Each consumer and producer conforms to its predicted profile, i.e., allocation and production rates are conformable to the profile. We plot in Figure 1 the total number of notification messages exchanged by allocators for different values of the divergence bound B.

The figure shows that the network traffic is quasi null when the divergence exceeds some limit (here 2000). This indicates that, for a reasonable value of bound B, there is no need for allocators to communicate when each one behaves conformably to the profile declared to consistency managers. If however the bound is too low, the network traffic remains important, even if there is no prediction error. A detailed analysis of the case of low bound

[3]Note that one could define a combined profile function $Prof_i$ as the sum of these two functions.

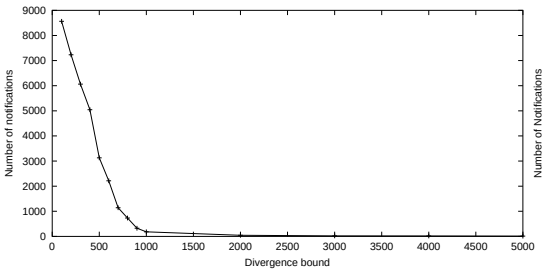

Figure 1: Network traffic in perfect case

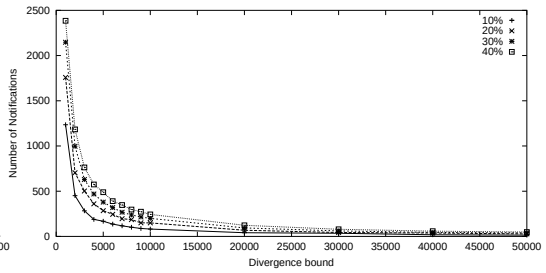

Figure 2: Network traffic with production miss

values reveals that approximatively 50% of allocation requests cause the error to exceed the local bound b_i. We conclude from this observation that our protocol is suitable when the application can afford large bound values. When bounds are too low, the required guarantee is almost the strong consistency, hence this becomes as costly as the strong consistency protocol.

In the second experiment, each producer generates resources with a frequency m% less than its announced frequency. The value of m indicates the prediction error. We plot in Figure 2 the resulting network traffic for various values of m.

First we note that the traffic decreases when the divergence bound increases. Therefore, for a fixed divergence bound the network traffic augments as the prediction error becomes higher. This clearly appears when divergence bounds are more restrictive i.e., low. But in the other hand the difference between network traffic for the different values of prediction error are not excessive for high bounds. Since the effect due to the producers behaviour is annihilated within the permitted divergence. This graph could be used to parameterize the system as follows: the system administrator fixes the network maximum traffic to generate and evaluates periodically the miss percentage of producers. Then he chooses the divergence bound accordingly to the observed network traffic.

5.2 Large Scale Settings

Here, we focus in evaluating the protocol in large scale settings. This includes large number of replicas, high requests rate and wide area network. For this purpose we build a virtual wide area network over our local network using a network emulator (dummynet [Riz97]).

We carried three experimentations to evaluate the protocol in large scale settings. First we studied the impact of replicas number on performance. We ran the application with different number of allocators on a local network. Each allocator serves a client that generates 50 requests per second. We plot in Figure 3 the response time versus the number of replicas. The response time remains between 1.5ms and 3.5ms. This is due to the local request treatment.

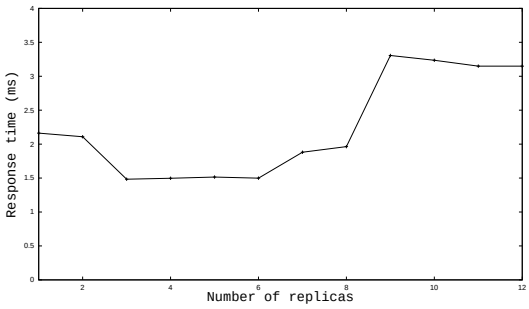

Figure 3: Response time

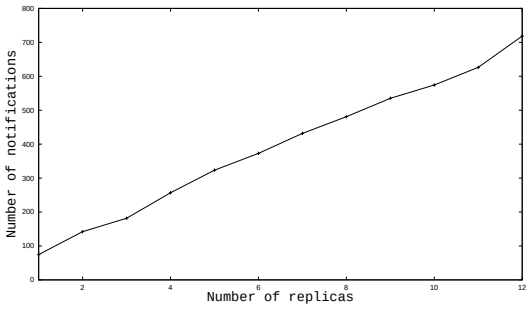

Figure 4: Network traffic

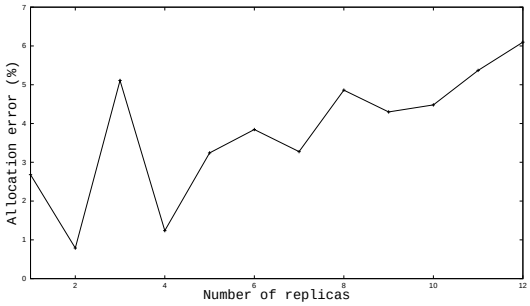

Figure 5: Allocation error

In Figure 4 we plot the corresponding number of notification messages exchanged between allocators. This figure shows that the generated network traffic is approximately linear with the number of servers.

The last interesting result in this evaluation case concerns application correctness and client satisfaction. As the protocol makes an acceptance decision on each request, we measured the client satisfaction as the correctness of this decision. Hence we plot in Figure 5 the difference between the number of allocated resources and the really available ones. Surely the protocol guarantees that this error remains smaller than the global bound B. Here we choose a bound corresponding to 10% of the number of all requested resources. This result is approximative (because we does not take care of occurrence date of allocation and production operations). We calculate this for each round and report the percentage of the error with respect to the number of requested resources. The graph shows that the error increases with the number of replicas, while remaining less than $B/2$.

In the second experiment, we run the group of allocators on a virtual network that simulates a wide-area round trip time. We studied two cases: the first with a RTT of 200ms and the second with a RTT of 500ms. Figures 6 and 7 plot the log_{10} of the number of synchronization messages corresponding to an increasing arrival request rate (up to 200

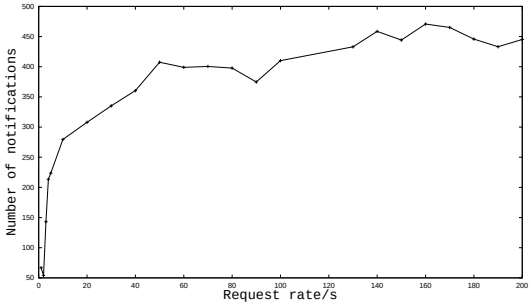

Figure 6: Network traffic with RTT=200ms

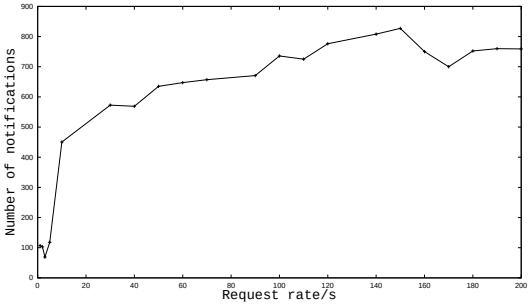

Figure 7: Network traffic with RTT=500ms

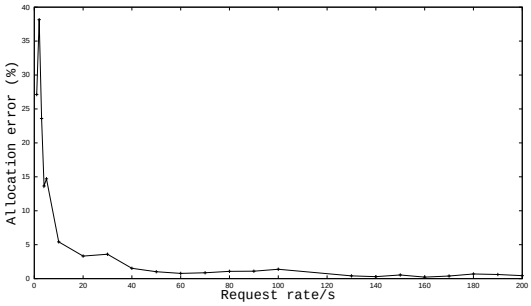

Figure 8: Allocation error with RTT=200ms

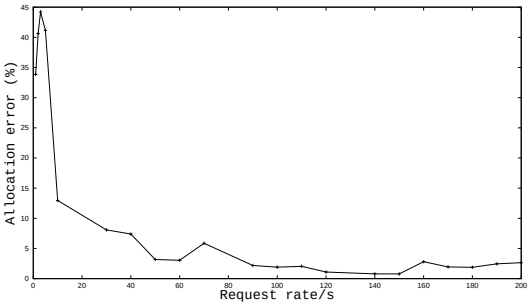

Figure 9: Allocation error with RTT=500ms

request/s). The two figures show that the network traffic generated by the protocol increases with the arrival request frequency and also becomes slightly stable with high frequencies.

As in the first case study, in order to evaluate the client satisfaction we plot in Figure 8 and Figure 9 the allocation error as a percentage of the all requested resources amount. These figures show that the protocol is much more interesting when used with high frequency.

5.3 Comparative Study

To evaluate the merit of divergence control protocol, we developed the evaluation application with two other protocols. The first is a periodic protocol : allocators periodically synchronize their replicas. At each synchronization time, remaining resources are equally distributed among all allocators. During a period each allocator serves its clients using its own part. When there is no more local resources, allocation requests are rejected. The second protocol implements a cooperative resources sharing. Allocator serves its clients using local production and query others when it does not have enough. This protocol principle is similar to ICP [ICP97], except the fact that it moves shared data between cooperative processes; i.e., resources exchanged are decreased from the sender and added to the receiver.

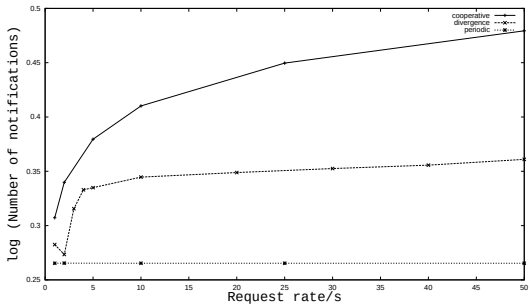

Figure 10: Comparative network traffic

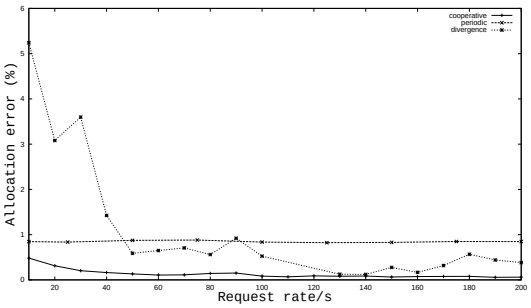

Figure 11: Comparative allocation error

We compared the network traffic generated by these protocols for the same request flow in a 200ms RTT network. Figure 10 shows that divergence control protocol saves network bandwidth compared to cooperative protocol. However, cooperative protocol makes less allocation error as it serves client using only existent resources (see Figure 11). Compared to the periodic protocol the divergence control protocol is more accurate with a slightly higher network traffic.

6 Discussion

6.1 Protocol Configuration

To run the protocol, we first need to determinate the bound B. In essence, this bound is used to model the risks that many real-world applications are confronted with. More formally, we refer to the divergence bound as an error window tolerated by the application. Hence, the suitable value of bound B depends on the application metric and on the error window that is considered as tolerable. Take the case of airlines reservation, overbooking flights is a common practice. This permits airlines companies to augment the occupancy of their flights. However, this practice has a cost, especially when companies are forced to refuse passengers that did book. To make sure that this cost won't exceed some fixed budget, one could fix the limit for overbooked seats per flight. This overbooking limit is a good candidate for bound value B.

Once the bound parameter is found, we need now to determine a tuple $(b_1, \ldots, b_n)$ for which condition 1 holds. One simple solution is to partition B into n equal parts ($b_i = \frac{B}{n}$). Note that this partitioning does not have to remain unchanged over the time. One could at any time adapt the partitioning provided that the sum of all parts remains less or equal to B. For the airline reservation example, we affect to each agency a part of B proportional to the total number of requests directed to this agency, compared to the total number of requests. Furthermore, one could increase or decrease the bound B. An increase of B augments the

degree of liberty of each cooperative server, while a decrease forces them to synchronize more often.

6.2 Application Characteristics

The protocol is designed with certain targeted applications in mind. First, we are targeting cooperative applications made of groups of cooperative processes deployed on geographically separate sites. We anticipate that the common state shared by these processes could be encapsulated by a well defined abstract object. Each cooperating process holds a local copy of the shared object. The shared object is fully determined by its initial state and the sequence of operations that application processes initiate on it; consequently re-executing the same operation sequence from the same initial state will produce an identical object state. Operations initiated on the object have two properties: (1) measurable, i.e., one can define for each operation a numerical value that indicates its effect on the shared object and (2) predictable, i.e., one can extract from application historic an evolution model using any existing monitoring and statistics techniques. Historic analysis can be based on traces formerly collected or on in fly traffic capture. Finally targeted applications correctness is not strongly related to the accuracy of replicas states, thus we exclude applications relying on strong consistency guarantees.

7 Related Work

In the area of weak consistency, several protocols have been proposed. Bayou proposes an optimistic replication protocol that addresses the specific needs of collaborative editing in a disconnected environment [TTP+95]. Bayou proposes an anti-entropy protocol for propagating updates. For each update a certification procedure and a reconciliation procedure are performed. It implements session guarantees to maintain consistency for clients switching from a replica to another. Like Bayou, our proposal is adapted for the cooperation between disconnected replicas. Unlike Bayou, we want to guarantee a bound on the divergence observed on each process.

Epsilon serializability [PL90, PL91] implements a transaction model on queries and updates. Each query or update introduces an inconsistency in the data. Two inconsistency accumulators are associated with each epsilon transaction: the first indicates the total amount of imported inconsistency, while the second indicates the inconsistency exported by the transaction. A transaction is not performed when those accumulators exceed a certain limit.

Alonso and al. propose the concept of "quasi-copy" caching [ABM90]. Updates are propagated to cached copies according to specified coherency conditions. Four types coherency conditions are proposed: arithmetic, delay, periodic and version. These conditions define the divergence that is permitted between the primary copy of an object and cached copies. These conditions can be implemented in our framework provided that one can define the suitable measure functions and can predict the primary copy access profile.

The formulation of consistency constraints in the demarcation protocol [BGM94] has some resemblance to the way we express our consistency criterion. Our protocol differs from the demarcation protocol in the way each replica determines its degree of liberty. In our case, replicas rely on their knowledge of the expected behaviour of others and on their allowable divergence set at the start time, whereas in the demarcation protocol a replica has to ask other replicas to change its allowable divergence.

Krishnakumar and Bernstein N-ignorance protocol [KB94] controls the divergence between database replicas by bounding to N the number of conflicting transactions of the same type that the system is allowed to run in parallel on different replicas. N is an inconsistency bound definition. This concept does not capture the effective impact of transactions that are concurrently initiated on different replicas. Our protocol is interested to bound that effect rather than the number of transactions.

In IceCube [KRSD01] divergence is allowed but a reconciliation is performed to obtain consistent states. IceCube is a log-based system. It runs operations over four stages: isolation, scheduling, simulation and selection. The outputed log is then reconciled. Because relying on operation constraints (dependencies), its combinatorial complexity could be expensive.

Our divergence control protocol is closely related to the TACT error bounding algorithm [YV00b, YV00a]. We attach a measure to each operation, which is similar to the notion of weight in TACT. However, relying on profiles allows to maintain the bound, even when replicas can not communicate. This might lead to a degradation of the service, rather than the denial of the service due to the inaccessibility of some partner.

The notion of eventually-serializable data service was introduced in [FGL+99] and implemented by [Che97]. Such a service maintains a partial order on operations that converges to a total order. This protocol is suitable for naming and directory service and for distributed information repository. Unlike our framework operations are of two types: "strict" operations must be ordered before client response and "non-strict" operations may be reordered after response time.

8 Conclusion

For applications making decision on uncertain knowledge, it is interesting that the imprecision on data they use is bounded. In a replicated context and in large scale settings, divergence control protocols are of primary interest for these applications. We presented here a divergence control protocol that benefits from replicas evolution profiles. The protocol controls on each replica the effect of the unexpected operations and ensures that this remains smaller than a given local bound. Performance analysis shows good performance in large scale networks including replicas number, high request frequency and round trip time. Current work aims to use this protocol with a disk space allocation service.

References

[ABM90] Rafael Alonso, Daniel Barbara, and Hector Garcia Molina. Data caching issues in an information retrieval system. *ACM Transactions on Database Systems*, 15(3):359–384, 1990.

[BC98] Georges Brun-Cottan. *Cohérence de données répliquées partagées par un groupe de processus coopérant à distance.* Thèse de doctorat, Université Paris 6, Pierre et Marie Curie, Paris (France), September 1998.

[BGM94] Daniel Barbara and Hector Garcia-Molina. The demarcation protocol: A technique for maintaining constraints in distributed database systems. *The International Journal on Very Large Data Bases*, 3(3):325–355, 1994.

[BHG87] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987.

[Che97] Oleg Cheiner. *Implementation and evaluation of an eventually-serializable data service.* Master 's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, September 1997.

[FGL+99] Alan Feket, David Gupta, Victor Luchangco, Nancy Lynch, and Alex Shvartsman. Eventually-serializable data service. *Theoretical Computer Science on Distributed Algorithms (Special Issue)*, 220(1):113–156, 1999.

[Hay98] Mark Hayden. The ensemble system. Technical Report TR98-1662, Cornell University, January 1998.

[ICP97] Internet cache protocol (ICP), version 2. RFC 2186, September 1997.

[JM01a] Ahmed Jebali and Mesaac Makpangou. Replica divergence control protocol in weakly connected environment. *Rapport de recherche*, June 2001. http://www.inria.fr/rrrt/rr-4217.html.

[JM01b] Ahmed Jebali and Mesaac Makpangou. Replica divergence control protocol in weakly connected environment. In *The IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, October 2001.

[KB94] Narayanan Krishnakumar and Athur Bernstein. Bounded ignorance: A technique for increasing concurrency in replicated system. *ACM Transaction on Data Base Systems*, 19(4):586 – 625, February 1994.

[KRSD01] Anne-Marie Kermarrec, Antony Rowstron, Marc Shapiro, and Peter Druschel. The icecube approach to the reconciliation of diverging replicas. In *the 20th annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, Newport, Rhode Island, USA, August 2001.

[LLSG92] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Transactions on Database Systems*, 10(4):360–391, 1992.

[PL90] Calton Pu and Avraham Leff. Epsilon-serialisability. Technical Report CUCS-054-90, Columbia University, 1990.

[PL91] Calton Pu and Avraham Leff. Replica control in distributed system, an asynchronous approach. In *the 1991 International conference on the managment of Data, ACM SIGMOD Record*, pages 377–386, Denver, Co, USA, May 1991. ACM SIGOPS, ACM press.

[PST+97] Karin Petersen, Mike J. Spreitzer, Douglas B. Terry, Marvin M. Theimer, and Alan J. Demers. Flexible update propagation for weakly consistent replication. In *16th ACM Symposium on Operating Systems Principles*, Saint Malo, France, October 1997.

[Riz97] Luidgi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, January 1997.

[TTP+95] Douglas B. Terry, Marvin M. Theimer, Karin Petersen, Alan J. Demers, Mike J. Spreitzer, and Carl H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. *the 15th ACM symposium on Operating System Principles, Operating System review*, 5(29):172–183, December 1995.

[Vap98] Vladimir Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[YV00a] Haifeng Yu and Amin Vahdat. Design and evaluation of a continuous consistency model for replicated services. In *Fourth Symposium on Operating Systems Design and Implementation*, October 2000.

[YV00b] Haifeng Yu and Amin Vahdat. Efficient numerical bounding for replicated network services. In *the 26th International Conference on Very Large Databases (VLDB)*, September 2000.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399